Educational Ruminations

Peter J. Denning

August 23, 1984

Research Institute for Advanced Computer Science
NASA Ames Research Center

RIACS TR 84 8

(NASA-CR-187279) EDUCATIONAL RUMINATIONS (Research Inst. for Advanced Computer Science) 11 p

N90-71403

Unclas
00/81 0295374

RIACS

Research Institute for Advanced Computer Science

Educational Ruminations

Peter J. Denning

July 25, 1984
Revision 1 August 13, 1984
Revision 2 August 23, 1984
Revision 3 August 27, 1984

Six times in twelve years, the heads of the PhD-granting computer science and engineering departments in the U.S. and Canada have met at Snowbird, Utah, to discuss common problems. Since 1980, they have focused much attention on the problem of attracting and retaining faculty, which is closely related to the problem of finding adequate resources to pay salaries and maintain experimental laboratories. The following remarks are based on my invited opening address at the Snowbird 84 meeting (July 30-31, 1984) and on the ensuing discussion.†

---

†I I am especially grateful to R. Ashenhurst, A. Borodin, D. Denning, and G. S. Graham for their comments on this material.

PAGE 1 INTENTIONALLY BLANK

In 1979-81 a series of reports, including the Feldman Report to NSF [*Communications ACM*, September 1979] and the Snowbird 80 report from the chairmen of the PhD-granting departments in computer science and engineering [*Communications ACM*, June 1981], helped pursuade the NSF and many university administrations that the "computer science problem" was real and helped bring relief in the form of badly-needed resources. Now that the needed resources have started to flow, it is time for the faculty once again to seize the initiative and deal with problems that are within their power to influence. In the past we asked, "What can others do for us?" Now we must ask, "What can we do for ourselves?"

We *are* receiving new resources, we *are* hiring new faculty, we *are* retaining faculty, we *are* seeing the teaching load reduced, we *are* getting the needed laboratory equipment, and most important we *are* taking on exciting new challenges in computer science research. The solutions to most of the remaining problems must come not from outsiders but from ourselves.

My purpose here is to suggest what some of the remaining problems are and, where possible, outline approaches to solutions. The areas are:

Salaries
Equipment and Facilities
Promotion and Tenure
Special Treatment of Junior Faculty
Long Range Planning
Core Curriculum
Relations with Other Disciplines
Nature of Research

The following sections look at each one of these issues in turn. The first four issues are familiar -- I simply want to remind ourselves that we have not fully resolved them.

1. Salaries

For new PhDs, the salary gap between academia and industry is narrowing and in many departments no longer exists. It is not unusual to find industrial offers for new PhDs of $48K this year (higher with certain types of experience); some of the larger departments have made offers of $36K for the first academic year plus 33% for the first summer.†

†A salary survey conducted at Snowbird 84 among 62 respondents showed these median 9-month salaries in the large departments (18 or more faculty): new PhDs $34,000; Assistant Professors with three years' experience $36,000; Associate Professors upon promotion $39,500. In the smaller departments, the corresponding figures were about $1000 lower.

To achieve these new-PhD salaries, many departments are tolerating "salary inversions," i.e., new salaries are close to senior-faculty salaries. This is an unstable situation. It will lead to discontent among the senior members of the faculty, who will then be susceptible to being recruited by other institutions. Departments with this problem should make every effort to eliminate it.

2. Equipment and Facilities

The Snowbird 80 report stated that a research lab capable of advancing the frontier of our science requires capitalization on the order of $50K to $75K per researcher. An analysis today of the laboratories in the major research departments leads to the same conclusion. These figures are similar to those in the other experimental disciplines such as physics, chemistry, or biology.

Less attention has been paid to laboratories for undergraduate instruction. These laboratories are essential to the integration of experimental computer science into the core curriculum. Fortunately, many industries recognize this and are offering to provide large donations of equipment for undergraduate laboratories.

To be successful with our laboratories, we must obtain sufficient floor space and provide a sufficient staff of technicians, programmers, and facilities managers to handle installation, reconfiguration, and maintenance. This staff must be capable of dealing with the highly heterogeneous systems and networks that will inevitably appear in our laboratories. Although the support staff for research is counted in the capitalization cost reported above, the support staff for instructional laboratories is an additional cost. The resources for this "laboratory infrastructure" must ultimately be provided by the university because government funds are generally restricted to specific research projects or are temporary, seed funds.

Because many outsiders have doubts about the permanence or depth of computer science, it is in many universities difficult to pursuade administrations to allocate the substantial new funds needed for our laboratories. (I will give a more complete analysis of these doubts in later sections.) We need more compelling arguments than "burgeoning student enrollments." Once outsiders are convinced of the staying power and substance of our discipline, they are more likely to be pursuaded by analogies between ours and other experimental disciplines.

3. Promotion and Tenure

A number of difficulties have arisen in recent years in connection with our system of promotion and tenure. Most universities abide by the AAUP guideline, which states that tenure must be granted within six years and that every

untenured faculty member must be given at least one year's notice of termination. (Taken together, these two guidelines force many departments to make a promotion decision in Year 5.) Most departments state in their promotions regulations that promotion and tenure will be awarded only if the individual has made contributions in research, teaching, and service, with significant contributions in at least one of these three areas.

A strong demand for personnel in a field generates distortions and difficulties with respect to the AAUP guidelines. In fields where PhD production is adequate, there is no rush to promote anyone. Many new PhDs take postdoctoral research positions for periods up to four years prior to joining the faculty, giving them as many as nine or ten years past the PhD to establish themselves as researchers. In the high-demand fields, there is a rush to promote – if your institution does not promote someone, another will. Because of this, many young faculty in our field *expect* early promotion and some threaten to go elsewhere if their departments do not take up their cases sufficiently early. Amid these tensions I see these difficulties:

1. Paradoxically, the apparent rigidity of the guidelines has produced a perception among graduate students that job security for assistant professors is low, which is one of the factors that discourages them from considering university careers. This perception exists even though many who choose industrial careers will switch jobs within six years anyway. (The difference is that the latter transition is voluntary.)
2. In spite of their own regulations on promotions, many departments put most of the emphasis on research and ignore teaching and service except when there is negative evidence. The young faculty member fears a normal teaching load not because he does not like teaching (most do) but because he feels he will not be rewarded as much for each hour of teaching as for each hour of research. Rather than reevaluate their practices for evaluating young faculty, many departments have been moving to set up special privileges for young faculty, which have been causing other distortions. (I will return to this later).
3. Many departments require research to have "significance and impact." Indeed! It is nearly impossible for a young faculty member's research to have measurable impact in time for the decision point in Year 5 – most reputable journals take upwards of two years for the referee-print cycle and most good ideas take five to ten years after that to influence the direction of the field. Many departments attempt to assess potential impact by reading the papers themselves and soliciting outside reviews but few study citation indices and other objective measures of "impact." Yet much of the research rewarded by promotion or tenure is mediocre and unlikely to have any impact.
4. Many department heads do not know how to argue tenure cases before their university committees. The unstated rule of the game is to demonstrate "peer recognition" and most arguments therefore rely on the publications lists and outside letters. This has given an edge to

cases of theoreticians. In fact, "systems oriented" and "experimentally oriented" faculty can be highly regarded by their peers because they distribute top quality software, they construct unusually fine computer systems, they attract grant money, they give excellent talks and receive many colloquium invitations, they attract good graduate students, etc. These other forms of peer recognition are excellent arguments for promotion but are often overlooked.

Solutions to these problems are difficult but not impossible. For example, the concept of "peer recognition" can be clarified for experimental scientists, leading to sound arguments for promotion from within this group. Departments should not be afraid to reward excellent teaching and excellent service more readily than mediocre research; they can, among other things be willing to give release time for curriculum development, textbook writing, and departmental service. In general, departments should not hesitate to reward excellence wherever it appears. They should avoid basing tenure cases on mediocre research.

4. Special Treatment of Junior Faculty

Many departments are establishing special privileges for junior faculty, including reduced teaching loads, guarantees to include graduate research seminars in the assigned courses, personal equipment funds, and in some cases special salary supplements. The argument is that the young researcher should be given every opportunity to develop into a mature researcher and a full teaching load and service assignment will block this development. In effect, we are trying to overlap a research postdoctoral appointment with an assistant professor appointment.

The administrative problem is that someone has to teach the students and staff the committees. So we often find these privileges being granted at the expense of the research time of the senior faculty, who become understandably resentful. This problem is exacerbated by the shortage of senior faculty -- in many computer science departments half the faculty are untenured whereas in most other disciplines the untenured fraction is much smaller.

There is nothing wrong with the argument that a young researcher needs to be given a chance to get established. The problems arise when this argument is carried to an extreme. (An amusing example of an extreme is the department who last year advertised a "Distinguished Assistant Professor Chair" to a new PhD -- presumably the distinction arose from holding the chair, not from any prior accomplishments.) When carried to extremes, the special treatment of junior faculty amounts to an implicit statement that the department values research but not teaching and service. This is most unfortunate because the best researchers are often excellent teachers.

In seeking balance, departments should aim to reward excellence in teaching and service as well as research. Special incentives should replace special privileges. A department can offer reduced loads for a period to any faculty member who can make a good case that he can accomplish something that will benefit the department. This can apply to teaching and service as well as research. For example, a reduced teaching load can be granted for course development, textbook writing, or heavy administrative duties.

5. Long Range Planning

Many department heads complain that their administrations have responded only in part to their pleas for increased resources and that their administrations simply "don't understand" what computer science is all about. In many cases the problem is not with the administration but with the department.

Many departments would find it rewarding to develop a written long range plan. The purpose of such a document is to set forth the goals of the department, its visions of the teaching and research environment at key points during the plan's period, and the personnel and resources required to achieve these goals.

A group of us did precisely that at Purdue in 1981. We discussed our visions of the classrooms, laboratories, and research environment by 1986 and 1989. We proposed specific limits on student enrollments and specific targets for staff size so that the teaching loads would be comparable to other departments in the School of Science. We constructed the organizational chart for the department's support staff in 1986 and 1989. We then converted these requirements into acquisition plans for faculty, supporting staff, equipment, and space. We were sufficiently specific that we could put a price tag on the whole project.

The result? Working with our dean, we formulated written arguments to overcome the most common objections to granting computer science more resources. Our written plan became the basis for a fund-raising drive and Purdue is now renovating a building for computer science. This building and its associated labs will provide Purdue with the resources to achieve its goals in computer science beginning in 1985.

6. Core Curriculum

The core curriculum of most CS departments is essentially a sequence of programming courses. The argument for this was first enunciated in the ACM Curriculum 68: Programming is at the foundation of everything we do in computer science. Once a student has mastered a core of programming courses, he

may opt for specialties such as business data processing, systems and architecture, scientific programming, or artificial intelligence. The curriculum for each specialty is a set of courses often structured as electives.

I believe this model of the CS core curriculum has come to the end of its useful life. Some of the indications of this are based on widely held outside perceptions of our field:

1. Many students from outside the department wish to enroll in the CS core courses to learn programming. They and their advisors cannot distinguish between "service programming courses" and "core programming courses" except that the latter are deeper and more challenging.
2. Scientists from other disciplines are increasingly curious about the substance of computer science. When they read our catalogs' descriptions of the core courses, they perceive little difference from what their lab technicians do. Their skepticism toward computer science remains high.
3. Ask a dozen computer scientists for a one-sentence definition of computer science. You will get a dozen different answers ranging from "study of algorithms" to "management of complexity" to "discrete problem solving."
4. Members of industry frequently criticize computer science curricula for being out of date. James Martin, for example, says that most graduates of CS departments know little or nothing about "fourth generation" concepts such as relational databases, structured design metholodology, distributed computation over networks, or logic programming. He asks not for "training" but for a solid intellectual foundation in these areas. While he agrees that teaching these concepts would be easier in departments with good experimental computer science laboratories, he argues that a major restructuring of the curriculum is needed so that these, and future developments, can find their natural places.

I conclude that computer scientists have no clear picture of the nature of their own field, which leads those from other disciplines to confused perceptions about us. We are projecting an illusion that we are mostly technicians and that our field has nowhere the same intellectual depth as the physical sciences or engineering. Computer science stands alone among science and engineering disciplines: Our curriculum has the technology in the core courses and the science in the electives!

I believe the time has come to seriously reexamine our approach to the core curriculum. In the process we can come to understand our own discipline better. Our core curriculum must be a clear statement, to ourselves and to outsiders, of the nature of our discipline.

In most physical sciences and engineering, the core curriculum is regarded as a *survey and tutorial of the discipline.* Its goal is to provide the student with a

solid conceptual framework, sharp analytic skills, and an ability to communicate ideas effectively. Course design clearly separates intellectual content (the core courses) from the technology (the lab courses). The typical pattern is three or four three-credit-hour courses, in which the students encounter every important concept of the discipline. Associated with each core course is a one-credit-hour lab (that meets for three hours weekly) in which the students conduct experiments illustrating the concepts covered in class. The first lab is tightly supervised; students are given detailed instructions on how to carry out each experiment and the types of results expected. The later labs are much less structured; students are given problem statements and are expected, with advice from lab instructors, to design and carry out experiments that solve the problem.

To apply this model in our discipline, we need to begin with a list of the principal areas of computer science in which we have discovered fundamental concepts. These certainly include:

Algorithms
Applicable Discrete Mathematics
Artificial Intelligence
Computer Communications
Complexity of Computation
Computer Architecture
Data Structures
Database Systems
Operating Systems
Programming Languages
Parallel Computation
Scientific Computation

Next, an order of presentation of the concepts must be determined and the concepts assigned to the core courses. The associated labs must be designed to give direct experience with the concepts in the courses.

For example, Course 1 will almost certainly contain algorithms and data structures. It would include a study of the most efficient algorithms for important classes of problems. Lab 1 would have the students learn to use an operating system, editor, compiler, loader, and debugger to create implementations of algorithms covered in class. They would be closely supervised by lab instructors who would ensure that basic principles (such as loop invariants) are actually being used in the programs. Note that Course 1 may cover concepts such as sequencing, cases, iteration, and loop invariants, but it will not cover specific details of a programming language. Lab 1 will cover specific details of a programming language and the operating system with which the students interact.

A later course will cover the fundamental concepts of operating systems; in the associated lab, the students can set up a network connection between two computers and measure the performance of protocols covered in class. A later course will cover the concepts of cooperating sequential processes; in the associated lab, students can observe race conditions and other anomalies resulting

from improper synchronization. Many other examples of this type exist.

The important feature of this model is that the core courses deal solely with fundamental concepts and their history. The detailed, experimental, "hands-on," technology-dependent aspects of the field show up in the labs. The labs are every bit as important in a student's education as the core courses, but are separate and distinct.

As in other sciences and engineering, a substantial investment in laboratories is required to support this type of core curriculum.

An interesting aspect of this model of a core curriculum is that it can be implemented without significant impact on the elective structure of the remaining curriculum. The electives are the places where students dig deeply into subjects covered in the core.

7. Relations with Other Disciplines

I have hinted above that computer science does not enjoy the respect of other sciences or engineering as a peer. Many physical scientists have no clear picture of our conceptual base and suspect it is shallow. Many engineers regard us as a field of mathematicians and programmers who have little appreciation for reality. Many outsiders perceive us as a field of technicians and "hackers" and they wonder whether our discipline will eventually be absorbed back into its progenitors.

There is a fairly clear strategy to overcoming the poor relations with the physical sciences. We need to develop a clear statement of our conceptual basis and incorporate it into a model of core curriculum closer to the one they use. We need to enter into interdisciplinary research projects where the research team includes computer scientists and engineers and works on fundamental questions in other disciplines. In my limited experience at the NASA Ames Research Center, where most projects are interdisciplinary, I have been very encouraged to discover the high degrees of mutual respect that develop in multidisciplinary projects. Much can be achieved by a less parochial attitude toward the other sciences.

Overcoming the problem with engineering may be more difficult. One complicating factor is the engineering disciplines' traditional desire for a high degree of autonomy and self-sufficiency. Thus they prefer to teach their own physics courses for solid-state engineering, their own chemistry courses for chemical engineering, and their own applied math courses for general engineering rather than ask the Physics, Chemistry, or Mathematics Departments to do these things. There have always been turf battles between the sciences and engineering; skirmishes over computer science are no exception. Moreover, because computing is an integral part of many engineering disciplines, it is fruitless to tell engineers that computer science must be separated from engineering.

Another complicating factor is that the country's top engineers are often not associated with any university. These are the engineers who deal with the design of real products, real production lines and manufacturing processes, real issues of quality control and human interface, real complexity and uncertainty, and real tradeoffs. Many students do not come into contact with these engineers or their experiences. One effect of these factors is that the research in engineering departments and computer science departments is strongly similar – a blend of applied science and theoretical engineering. Many industrial observers do not think computer science and engineering departments are pushing the frontiers of research. (For example, many departments are now undertaking projects to develop window packages for workstations, even though within a year there will be at least a dozen commercial workstations offering window packages backed by several years of engineering.)

So, to achieve greater respect among engineers, computer scientists need to stop arguing for separation and to start seeking closer working contacts with industry. At the university level, we must work for policies that allow cooperative joint ventures with industry. At the department level, we must allow for teaching about the technology without giving the impression it is part of the intellectual core.

8. Nature of Research

I suggested above that much computer science research is a blend of applied science and theoretical engineering that does not reach out very far into the physical sciences or real engineering.

With better contact in the computer manufacturing industry, we would see more research on circuit design, "silicon compilers", better instrumentation for controlling microcircuit manufacturing lines, better methods for testing integrated circuits, better methods of building reliable machines, robotics, automatic manufacturing systems, and the like. With better contact in the sciences, we would see more research on airflow simulation in three dimensions, calculating chemical properties of materials from basic principles, simulated laboratory experiments, simulation of basic processes underlying genetic engineering, astrophysical simulations, new algorithms for solving the open problems of critical phenomena throughout physics and chemistry, and the like. The research in the field would be much, much broader in its scope.

The pressures on young researchers to produce results within a short time is not helping this problem. Interdisciplinary projects can take a long time for the researcher to gain a critical mass of knowledge and begin making basic contributions. The mass of mediocre research being churned out by congeries of young researchers under duress to demonstrate promotability threatens to bury the high-quality output of our most creative researchers.